ADVANCES IN AUTONOMOUS SYSTEMS FOR SPACE EXPLORATION MISSIONS

By

Anthony R. Gross[1], Benjamin D. Smith[2], Daniel J. Clancy[1], Howard N. Cannon[1], Anthony Barrett[2],

Edward Mjolssness[2], Nicola Muscettola[1], Steve Chien[2], and Andrew Johnson[2],

[1]NASA-Ames Research Center, Moffett Field, CA, USA

[2]Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, USA

Abstract

System autonomy capabilities have made great strides in recent years, for both ground and space flight applications. Autonomous systems have flown on advanced spacecraft, providing new levels of spacecraft capability and mission safety. Such capabilities enable missions of such complexity and communications distances as are not otherwise possible, as well as many more efficient and low cost applications. This paper will focus on new and innovative software for remote, autonomous, space systems flight operations, including distributed autonomous systems, flight test results, and implications and directions for future systems. Topics to be presented will include a brief description of key autonomous control concepts, the Remote Agent program that commanded the Deep Space 1 spacecraft to new levels of system autonomy, recent advances in distributed autonomous system capabilities, and concepts for autonomous vehicle health management systems. A brief description of teaming spacecraft and rovers for complex exploration missions will also be provided. New software for autonomous science data acquisition for planetary exploration will be described, as well as advanced systems for safe planetary landings and an innovative personal satellite assistant concept. Finally, plans and directions for the future will be discussed.

Introduction

When Europeans first began to explore the New World, they depended heavily upon the intelligence of the explorers and their ability to utilize the resources available within the newly explored territory. Thus, when Lewis and Clark first embarked on their voyage across North America, they left knowing little about what they would encounter on their voyage. To accomplish the exploration goals that lay ahead of them, they depended heavily upon their own intelligence and ingenuity to respond to the circumstances encountered while utilizing available resources. These same capabilities have often proved beyond Earth. The successful safe return of the crew of Apollo 13 provides a compelling story of the ability to respond to unforeseen circumstances using the limited resources available at that time. Of course, the Apollo 13 incident demonstrated the ingenuity of both the crew and the large ground support team that worked around the clock to explore many different scenarios for the safe return of the crew to earth. Unfortunately, as we start to consider further manned exploration of our solar system, communication delays and budgetary constraints limit our ability to depend upon a large ground support team especially for the routine, day-to-day operation of the mission [1,2].

Robotic spacecraft are making it possible to explore the

.....

Presented at the Core Technologies for Space Systems Conference, Colorado Springs, CO, November 30, 2001.

* Anthony R. Gross is Associate Director of the Information Sciences and Technology Directorate; Benjamin D. Smith is Manager of the Autonomy Technology Program at JPL; Daniel J. Clancy is Chief (Acting) of the Computational Sciences Division, Howard N. Cannon is Project Manager for the NITEX Project; Anthony Barrett is a senior member of the AI Group; Eric Mjolssness is Supervisor of the Machine Learning Group; Nicola Muscettola is Lead for the Automation and Robotics Group; Steve Chien is Supervisor of the Artificial Intelligence Group.

other planets and understand the dynamics, composition, and history of the bodies that make up our solar system. These spacecraft enable us to extend our presence into space at a fraction of the cost and risk associated with human exploration. They also pave the way for human exploration. Where human exploration is desired, robotic precursors can help identify and map candidate landing sites, find resources, and demonstrate experimental technologies.

Current spacecraft control technology relies heavily on a relatively large and highly skilled mission operations team that generates detailed time-ordered sequences of commands, or macros, to step the spacecraft through each desired activity. Each sequence is carefully constructed in such a way as to ensure that all known operational constraints are satisfied. The autonomy of the spacecraft is therefore quite limited.

With exponentially increasing capabilities of computer hardware and software, including networks and communication systems, a new balance of work is being developed between humans and machines. This new balance holds the promise of greatly increased space exploration capability, along with dramatically reduced design, development, test, and operating costs. New information technologies, which take advantage of knowledge-based software and high performance computer systems, will enable the development of a new generation of design and development tools, schedulers, and vehicle and system health monitoring capabilities. Such tools will provide a degree of machine intelligence and associated autonomy which has previously been unavailable to the mission and spacecraft designer and to the system operator. These capabilities are critical as we look toward future exploration of our solar system due to both the requirements levied by these missions as well as the budgetary constraints that limit our ability to monitor and control these missions using a standing army of ground-based controllers.

In the next section, autonomous control techniques being developed at NASA are discussed. These concepts are being developed and demonstrated within the context of a broad range of mission scenarios, from singe spacecraft to teams of spacecraft and rovers that cooperate in the exploration tasks.

Autonomous Control Concepts

The ability to autonomously monitor and control complex devices such as a robotic explorer system is critical to NASA's ability to accomplish many of its long-term space exploration goals. From the beginning of the space program in the late 1950's, control of spacecraft and systems have been managed by a large number of highly trained ground control personnel. This has its roots in the limited capability and massive size of computers of that early period. In addition, sensor data was telemetered to the ground, where a room full of experts would monitor each individual system's health and send commands to the spacecraft, either directly or via an astronaut. Over the past forty years there has been a radical shift in this paradigm, resulting largely from the advent of advanced computer technology. Automation has eased the burden of the ground controller and the astronaut, but often the tasks performed by the software are still quite rudimentary. This results from both computational resource limitations and the difficulty encountered when trying to develop, test, and validate software that provides the required

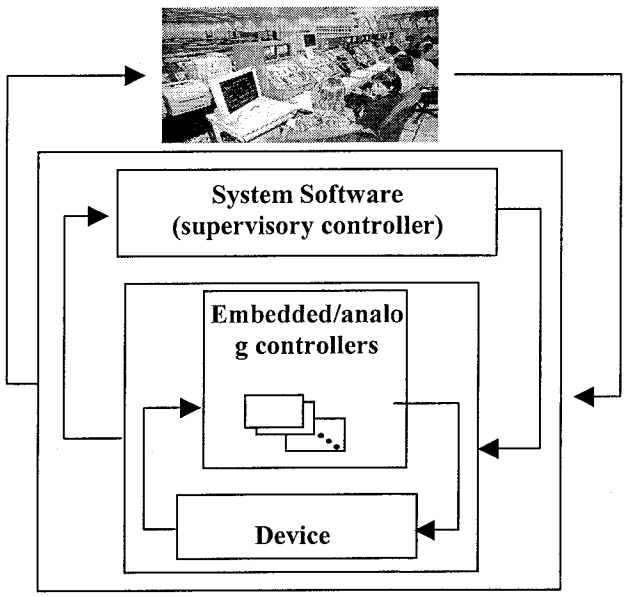

Figure 1: Tiered control architecture

functionality. As missions extend outward in the solar system, beyond the Earth-Moon system, the physical and fiscal realities of space exploration will require new control technologies.

Conceptually, the task of controlling a device such as a planetary rover is principally one of maintaining the system in a stable state while commanding transition of the device through a series of configurations designed to accomplish a sequence of goals in some optimal fashion. This task however, is often quite difficult due to the normal variations that occur within both the process and the environment, limited observability into the current state of the device, and the potential of abrupt failures and degraded component performance. Traditionally, these problems have been solved through the use of a tiered

architecture comprised of three levels, as shown in figure 1:

1. ***analog and embedded feedback controllers*** to perform low-level regulatory functions,
2. ***higher-level system software*** to perform nominal command sequencing and threshold monitoring to detect and respond to off-nominal conditions, and
3. ***humans*** to generate the command sequences, monitor the state of the device to detect off-nominal conditions, diagnose failures when they occur and select recovery actions in response to these failures.

While the capabilities provided by the system-level software have substantially increased over the past 40 years, the complexity of the missions undertaken by NASA have also increased. As a result, the requirements levied on the ground control team have increased, thus requiring larger ground support teams. This paradigm begins to break down for future missions, due both to greatly increasing time delays in communication as well as increasing costs of maintaining a larger ground support team. As a result, the role traditionally performed by the ground support team is being shifted to the system-level software, thereby greatly increasing the functionality required of this component. Figure 2 shows how the functionality provided by these three different components has shifted over the years, and where it is expected that the responsibility will lie when supporting a human expedition to Mars, for example.

Currently, the system-level software is developed by engineers who use commonsense understanding of the hardware and mission goals to produce computer code and control sequences that will allow a spacecraft, or other system, to achieve a particular goal while allowing for some (usually very small) amount of uncertainty in the environment. In developing this code, the engineer must reason through complex sub-system interactions to generate procedural code that can account for all the different combinations of failures and off-nominal conditions that might occur. As the functionality that is required of the system-level software increases, development, test, validation and maintenance of this software, using this traditional approach, becomes very difficult, if not impossible, as a result of the myriad of off-nominal conditions that the software is expected to handle. Furthermore, as the engineers gain a better understanding of how the device is behaving after deployment, it is often quite difficult to update the code to reflect the additional information that has been obtained.

Artificial Intelligence and Autonomous Control

As attempts are made to automate the processes that are traditionally performed by humans when monitoring and controlling these devices, it is clear that it will often be necessary to replicate the sophisticated inferencing capabilities exhibited by humans when performing this task. Over the last 4 decades, the field of artificial intelligence has been developing a variety of automated techniques that emulate a human's reasoning ability [3]. While the systems developed are far from performing at

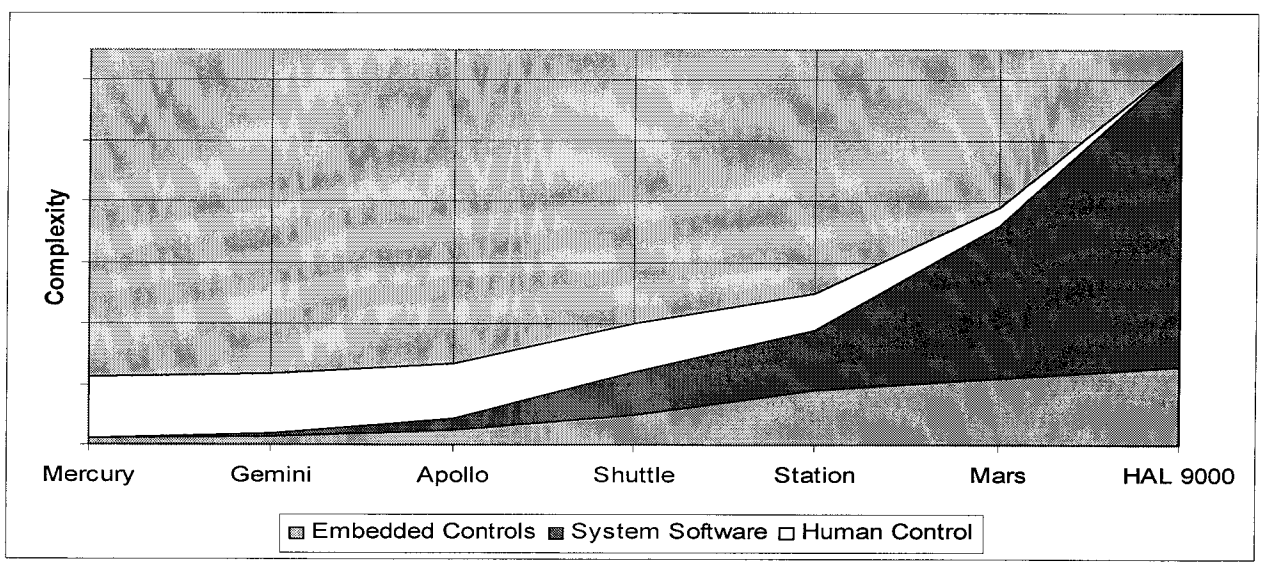

Figure 2: Progression across missions of the tasks performed by humans, system software and embedded controllers.

the visionary level exhibited by the HAL 9000 computer in *2001: A Space Odyssey,* such accomplishments as the victory of Deep Blue over Kasparov have demonstrated that sophisticated inferencing tasks can, indeed, be automated.

As an example of a first major step in providing an autonomous operating capability to an actual spacecraft, NASA developed and demonstrated, in flight, the Remote Agent (RA) autonomous control architecture. The next section will briefly describe that architecture and the subsequent flight experiment that led to a new space mission capability.

Remote Agent and the Deep Space 1 Mission

The Remote Agent architecture, developed collaboratively between NASA Ames Research Center (ARC) and the Jet Propulsion Lab (JPL), was part of the Deep Space One mission within the NASA New Millennium Program (NMP). It combined high-level planning and scheduling, robust multi-threaded execution, and model-based fault detection isolation and recovery, into an integrated architecture that was able to robustly control a spacecraft over long periods of time [4-9]. The overall architecture of the Remote Agent, along with the additional elements incorporated for the Deep Space 1 mission, is diagrammed in Figure 3, and will be explained below.

Remote Agent itself was made up of three major components, each of which played a significant, integral role in controlling the spacecraft: Planner and Scheduler (PS)—produces flexible plans, specifying the basic activities that must take place in order to accomplish the mission goals. Smart Executive (EXEC)—carries out the planned activities. Mode Identification and Recovery (MIR)—monitors the health of the spacecraft and attempts to correct any problems that occur.

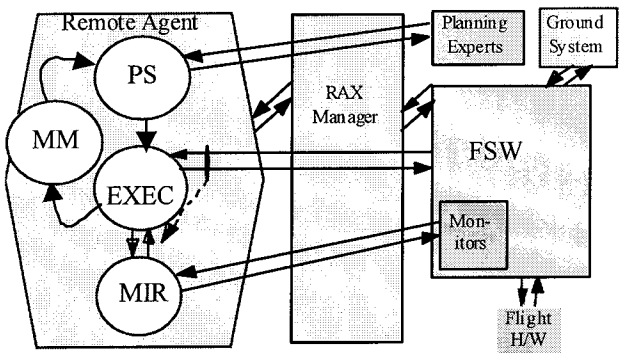

Figure 3. Remote Agent and Deep Space 1 Architecture

These three parts worked together and communicated with each other to make sure the spacecraft accomplished the goals of the mission: EXEC requested a plan from PS. PS produced a plan for a given time period based on the general mission goals and the current state of the spacecraft. EXEC received the plan from PS, filled in the details of the plans, (eg., determined what spacecraft system actions must take place to complete the planned activities), and commanded the spacecraft systems to take the necessary actions. MIR constantly monitored the state of the spacecraft. It identified failures and suggested recovery actions. EXEC executed the recovery action or requested a new plan from PS that would take into account the failure. The parts of Remote Agent were constantly communicating (using inter-process communication) with each other and with the external components of the spacecraft. MIR received information regarding the state of different components from monitors located throughout the spacecraft. PS must receive information from planning experts in order to generate the plan. For example, the navigation system reports to PS regarding the spacecraft's current position, and the attitude-control system tells PS how long it will take to turn the spacecraft to a new position. Finally, EXEC sends commands to other pieces of flight software that, in turn, control the spacecraft's systems or flight hardware.

As proof of the concept and capabilities, the Remote Agent software operated NASA's Deep Space 1 spacecraft and its futuristic ion engine during two experiments in May of 1999. For two days Remote Agent ran on the on-board computer of Deep Space 1, more than 60,000,000 miles (96,500,000 kilometers) from Earth. The tests were a successful step toward robotic explorers of the 21st century that will be less costly, more capable, and more independent from control from Earth.

Future exploration missions envision utilizing multiple rovers that are able to interact among themselves and thus greatly increase the science data return. The next section describes the research and development activities that are the basis of the capabilities required for such advanced missions.

Distributed Autonomous Systems

From the Terrestrial Planet Finder to robots helping each other scale cliffs on Mars, many future NASA mission concepts involve teams of tightly coordinated spacecraft/rovers in dynamic, partially understood environments. In order to maintain team coherence, each spacecraft must robustly respond to team coordination anomalies as well as local events. Currently manual techniques for implementing such teams are extremely

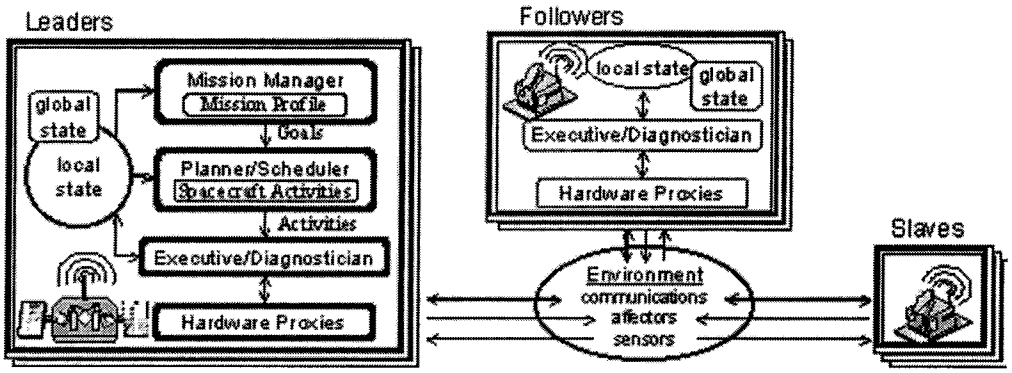

Figure 4. Software anatomy of leaders, followers, and slaves in an autonomous constellation.

difficult. These techniques involve either having one spacecraft tightly control the others or giving each spacecraft a separate activity sequence with explicit communication actions to coordinate with other spacecraft. While both approaches can handle two or three simple spacecraft, neither scales to larger populations or more complex spacecraft. New techniques are needed to facilitate managing populations of 3 or more complex spacecraft.

In general, autonomous spacecraft and rovers must balance long-term and short-term considerations. They must perform purposeful activities that ensure long-term science and engineering goals are achieved and ensure that they each maintain positive resource margins. This requires planning in advance to avoid a series of shortsighted decisions that can lead to failure. However, they must also respond in a timely fashion to a dynamic and partially-understood environment. In terms of high-level, goal-oriented activity, the spacecraft must modify their collective plans in the event of fortuitous events such as detecting scientific opportunities like a sub-storm onset in Earth's magnetosphere or a Martian hydrothermal vent, and setbacks such as a spacecraft losing attitude control. Whether they are orbiters, probes or rovers, coordinating multiple distributed agents introduces unique challenges. Issues arise concerning interfaces between agents, communication bandwidth, group command and control, and onboard capabilities. For example, consider a mission with a cluster of satellites simultaneously observing a point on a planet from different angles with different sensors. A certain level of communication capabilities will need to be assigned to each, possibly limiting the amount of information that can be shared between the satellites (and a ground station). The onboard capabilities also need consideration, including computing power and onboard data storage capacity. This will limit the level of autonomy each of the satellites can have. Finally, these issues apply to multiple rover missions too. A group of rovers might want to simultaneously measure vibrations caused by an explosion to determine the subsurface geology of an area on Mars. Developments are currently underway in team planning and execution to address these issues [10].

Team Plans

Current missions control multiple spacecraft by either giving one spacecraft a control sequence and having it treat the others as if they were virtually connected or giving each spacecraft its own command sequence with inserted communications actions for coordination between spacecraft. Neither approach scales as populations increase nor as members become more capable. While the first requires too much bandwidth, the second suffers from instabilities that lead to a variety of coordination failures.

These limitations can be overcome with hierarchical 'team plans' [11], which provide a rich representation for coordinated actions and communication while allowing the flexibility needed to adapt those plans in response to execution-time uncertainties. The representation for reactive team plans assumes a model of teamwork implemented on agents (like spacecraft) whose architectures explicitly support representing and reasoning about team goals, team plans, and team states. Team plans are hierarchical plans generalized to include team operators. Our approach builds on hierarchical reactive plans to provide a representation for explicitly defining constellation behavior. Giving each spacecraft the full constellation plan has several advantages over breaking up a plan into different role plans for each spacecraft. It enables a spacecraft's awareness of how its current activity relates to other spacecrafts' activities. This information makes a constellation more robust by letting spacecraft monitor each other's progress. Also, this information facilitates autonomous recovery when

unexpected events happen. When one spacecraft fails to make progress, the related spacecraft can coordinate their responses.

Onboard Team Planning

While team plans provide a powerful way to specify team behaviors, teams operating in uncertain environments must be able to adapt those plans in response to unexpected events and opportunities. This capability is provided by onboard team planning, which extends single-spacecraft onboard planning to address issues of coordination and limited inter-agent communication bandwidth.

In order to facilitate this form of goal-based constellation commanding we are developing two complementary techniques. *Goal Distribution Planning* will let a designated lead spacecraft/rover plan with an abstract model of all followers in order to assign goals across the population. *Contract Networks* will let any spacecraft/rover serve as an auctioneer to distribute goals. These two approaches are actually two points in a spectrum of approaches where the leader gives its followers progressively more autonomy in deciding who satisfies which goals and how to satisfy a goal. While the first approach uses one spacecraft to collect all constellation information and generate a team plan/schedule, the second reduces communication overhead by spreading planning and scheduling activities across all spacecraft. In addition to extending these approaches to work with continual planners, this research explores the spectrum by developing a hybrid system that uses both approaches.

Goal Distribution Planning (Master/Slave)

In the master/slave approach, one lead spacecraft embodies all four of the components and teleoperates the others. Our approach extends the MADS master/slave approach toward teamwork. In this approach a distinguished lead spacecraft will collect goals, generate a fleet plan, and broadcast it for execution. Just like a self-commanding spacecraft, the lead constellation spacecraft must respond in a timely fashion in a dynamic partially understood environment.
While this approach benefits from conceptual simplicity, it relies on an assumption that the leader's hardware proxies can continuously monitor the slaves' hardware, and this relies on high-bandwidth highly reliable communications. Reducing the requirements involves localizing reactive feedback loops by putting hardware proxies on all spacecraft, but this requires replicating the executive/diagnostician to appropriately manage the local hardware proxies. The final result is a constellation control architecture where the constellation's leaders, followers, and slaves depend on mission requirements.

Contract Networks

As an alternative to the master/slave approach, we are also pursuing work on market-based approaches to constellation planning [12]. The master/slave approach places too much of a burden on the lead spacecraft by requiring it to know the full state of all constellation spacecraft prior to planning and scheduling. The market-based approach is one of a family of approaches that spreads the burden across the constellation by designing protocols and mechanisms through which each spacecraft acquires goals and resources to generate the desired team behavior. In a computational market, distributed agents interact with each other to exchange goods and services. With these protocols, agents participate in a computational economy by interacting in the market to further their own interests. This technique, called market-based programming, is built on the observation that techniques from economics can apply to protocol design to assure appropriate allocations of goals and resources for an efficiently functioning constellation.

In the market-based approach, the agents with mission goals (spacecraft) are defined as consumers. Agents performing activities are producers, and the preconditions and results of activities are goods. The lead (or any other) spacecraft can act as a consumer by advertising a mission goal with its value. Spacecraft bid for the work, and the winner would take the goal and act as a producer by performing an activity to resolve it and possibly subcontracting subgoals to other spacecraft to bid on.

Closed-loop Science Autonomy

Closed-loop science autonomy technologies are being developed that will enable scientists to prioritize and reduce data onboard in order to make the best use of limited communication bandwidth, to scan high-rate instrument data streams for short-lived or hard-to-find events, and to detect and exploit short-lived science opportunities that would otherwise be lost.

In current science missions ground-based science teams decide what observations to take based on information from Earth-based observations, prior missions, and even new data from the current mission. While this decision loop enables extremely high-quality decisions, it is also a very slow one. Closed-loop science autonomy migrates some of that intelligence onboard the spacecraft to enable on-the-spot decisions when there just is not enough time or bandwidth to make those decisions on the ground.

Two of the closed-loop science autonomy technologies being developed at JPL are described below. The first of these focuses on orbital missions, whereas the second focuses on landed missions.

Autonomous Sciencecraft Constellation

The Autonomous Sciencecraft Constellation flight demonstration (ASC) will fly onboard the Air Force's TechSat-21 constellation (an unclassified mission scheduled for launch in 2004). ASC will use onboard science analysis, replanning, robust execution, model-based estimation and control, and formation flying to radically increase science return by enabling intelligent downlink selection and autonomous retargeting.

ASC will autonomously recognize science opportunities and then reconfigure the constellation to acquire focused images on subsequent orbits. This onboard recognize-and-replan loop enables the spacecraft to dramatically increase the science per fixed downlink by enabling downlink of only the highest priority science data. Demonstration of these capabilities in a flight environment will open up tremendous new opportunities in planetary science, space physics, and earth science that would be unreachable without this technology. ASC is a collaboration including JPL, the Air Force Research Laboratory, Interface & Control Systems, Princeton Satellite Systems, Arizona State University, the University of Arizona, and MIT.

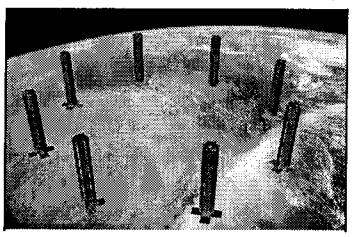

Figure 5: TechSat-21 (AFRL)

The ASC onboard flight software includes several autonomy software components:

- *Onboard science algorithms* that will analyze the image data, generate derived science products, and detect trigger conditions such as science events, "interesting" features, and change relative to previous observations (e.g., [13,14]).
- The Continuous Activity Planning, Scheduling, and Replanning (CASPER) [15] *planner* that will replan activities, including downlink, based on science observations in the previous orbit cycles.
- *Model-based mode identification and execution (MI-R)* that uses component-based hardware models to analyze anomalous situations and to generate novel command sequences and repairs.
- *Robust execution management software* using the Spacecraft Command Language (SCL) [16] package to enable event-driven processing and low-level autonomy.
- The ObjectAgent and TeamAgent *cluster management software* will enable the three Techsat-21 spacecraft to autonomously perform maneuvers and high precision formation flying to form a single virtual instrument.

The onboard science algorithms will analyze the images to extract static features and detect changes relative to previous observations. Prototype software has already been demonstrated on X-band radar data (from shuttle missions) to automatically identify regions of interest. This includes identification of regions where changes such as flooding, ice melt, and lava flows have occurred; and recognition of features such as craters and volcanoes. Such onboard science will enable retargeting and search, e.g., shifting the radar aim-point on the next orbit cycle to identify and capture the full extent of a flood. Onboard science analysis would also enable capture of short-lived science phenomena at the finest time-scales without overwhelming onboard caching or downlink capacities. Examples include of this include: eruption of volcanoes on Io, formation of jets on comets, and phase transitions in ring systems. Generation of derived science products (e.g., boundary descriptions, catalogs) and change-based triggering will also reduce data volumes to a manageable level for extended duration missions that study long-term phenomena such as atmospheric changes at Jupiter and flexing and cracking of the ice crust on Europa.

The onboard planner (CASPER) will generate mission operations plans from goals provided by the onboard science analysis module. The model-based planning algorithms will enable rapid response to a wide range of operations scenarios based on a deep model of spacecraft constraints, including faster recovery from spacecraft anomalies. The onboard planner will accept as inputs the science and engineering goals and ensure high-level goal-oriented behavior for the constellation.

The robust execution system (SCL) accepts the CASPER-derived plan as an input and expands the plan into low-level commands. SCL monitors the execution of the plan and has the flexibility and knowledge to perform event-driven commanding to enable local improvements in execution as well as local responses to anomalies. Livingstone 2 performs model-driven estimation of spacecraft state, and Burton [21] also accepts configuration goals from SCL. The ObjectAgent and TeamAgent cluster management software manages the maneuver planning and execution for the constellation. It accepts high-level constellation formation goals from CASPER and plans and executes these formations to support science (e.g. radar imaging) and engineering (e.g. downlink) activities.

One of the ASC demonstration scenarios involves monitoring of lava flows in Hawaii. SIR-C radar data have been used in ground-based analysis to study this phenomenon. The ASC concept would be applied as follows:

(1) Initially, ASC has a list of science targets to monitor.
(2) As part of normal operations, CASPER generates a plan to monitor the targets on this list by periodically imaging them with the radar.
(3) During such a plan, a spacecraft images the volcano with its radar.
(4) Onboard, a reflectivity image is formed.
(5) The *Onboard Science Software* compares the new image with previous image and detects that the lava field has changed due to a new flow. Based on this change the science software generates a goal to acquire a new high-resolution image of an area centered on the new flow.
(6) The addition of this goal to the current goal set triggers CASPER to modify the current operations plan to include numerous new activities in order to enable the new science observation. During this process CASPER interacts with ObjectAgent to plan required slews and/or maneuvers.
(7) SCL executes this plan in conjunction with several autonomy elements. Mode Identification assists by continuously providing an up to date picture of system state. Reconfiguration (Burton) achieves configurations requested by SCL. And ObjectAgent and TeamAgent execute maneuvers planned by CASPER and requested at run-time by SCL.
(8) Based on the science priority, imagery of identified "new flow" areas; are downlinked. This science priority could have been determined at the original event detection or based on subsequent onboard science analysis of the new image.

As demonstrated by this scenario, onboard science processing and spacecraft autonomy enable the focus of mission resources onto science events so that the most interesting science data is downlinked. In this case, a large number of high priority science targets can be monitored and only the most interesting science data (during times of change and focused on the areas of change) need be downlinked.

The ASC concept has been selected for flight on the Techsat-21 mission and the necessary software is currently being matured and brought into flight readiness. Key Techsat-21 design reviews occur in Spring 2001 to Spring 2002, with final delivery of the spacecraft and software in September 2003. Nominal launch date is September 2004. The NASA New Millennium Space Technology 6 Project has selected the ASC concept for a Phase-A award.

Hypothesis-directed Science

In many science acquisition scenarios it is important not only to detect science opportunities, but also to evaluate them in order to select the best subset for focused observations. This evaluation depends on the expected utility of the observation and the expected cost of acquiring it, relative to the other opportunities. The expected utility depends not only on a static 'figure of merit', but also the degree to which it is expected to support or weaken an emerging scientific hypothesis when combined with other observations. We refer to this as *hypothesis-directed* science.

The Multi-Rover Integrated Science Understanding System (MISUS) autonomously identifies and exploits geologic science opportunities in landed-mission scenarios [17]. It employs machine learning methods to identify, evaluate, and select science opportunities. It then employs a distributed planning component (ASPEN [15]) to determine the sequence of rover activities needed to acquire them and to estimate the acquisition cost based on those activities. Planning activities are distributed among the individual rovers where each rover is responsible for planning for its own activities. A central planning system is responsible for dividing up the goals among the individual rovers in a fashion that minimizes the total traversing time of all rovers.

Science data analysis in MISUS is performed using machine learning clustering methods, which use image and spectral mineralogical features to help classify different planetary rock types. Specifically, the clustering methods employed look for similarity classes of visual texture and reflectance spectra across multiple targets such as rocks. These clusters can then be used to help evaluate scientific hypotheses and also to prioritize visible surfaces for further observation based on their ``scientific interest." As the clusterer builds a model of the rock type distribution, it continuously assembles a new set of observation goals for a team of rovers to collect from different terrain locations. Thus, the clusterer drives the science process by analyzing the current data set and then deciding what new and interesting observations should be made. Clustering in MISUS is performed by a novel distributed algorithm where each rover alternates between independently performing learning computations using its local data and updating the system-wide model through communication among rovers.

Efforts are underway to incorporate recently developed hypothesis-directed algorithms for this evaluation and selection. These algorithms utilize the rock-type classifications and spatial clustering results to determine which geological processes were most likely to have formed the observed deposits. The observations provide evidence for or against various parameterized formation models (hypotheses), and for identifying the most likely parameter settings. Figure 5 shows one such parameterized formation model for crater-impact deposits. We have modeled the ejecta using the simple and phenomenological z-flow model for excavation augmented to include rock fragmentation by shock, and ballistic emplacement. The hypothesis-directed algorithms determine which parameter settings were most likely to have resulted in the observed rock types and spatial distribution.

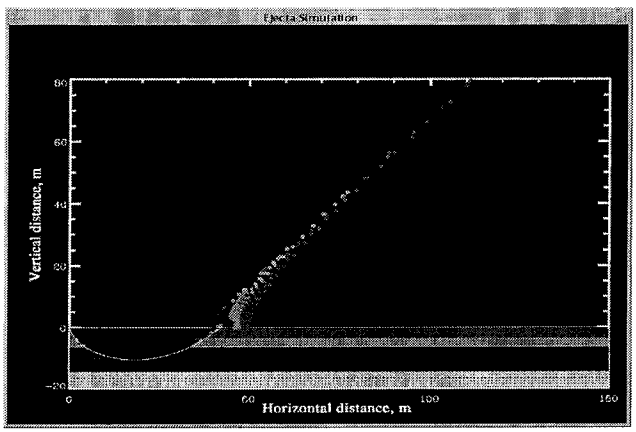

Figure 6. Model output of excavation flow from simple crater-forming impact.

This hypothesis information enables the system to select observations that are most likely to enable scientists to identify the geological processes and answer critical questions than they could from observations evaluated based on simple 'figure-of-merit' calculations. Hypothesis-directed capabilities will be particularly important for future missions, such as Titan Organics Explorer and the Europa Hydrobot, where infrequent communications and dynamic environments will require unprecedented levels of onboard science decision making.

These kinds of geological models are, we propose, essential to creating future intelligent *in situ* spacecraft software because they provide the foundation for evaluating the effects of new data on the likelihood of alternative geological hypotheses. Adapting pattern recognition techniques to infer such models from data is one route to intelligent, goal-directed robotic exploration.

Safe Landing Systems

Exploration of the solar system will require robotic systems that can navigate and land safely on planets and small bodies. Due to the small size, irregular shape and variable surface properties of small bodies, accurate position estimation and hazard avoidance are needed for safe and precise small body landing and sample return. Because of the communication delay induced by the large distances between the earth and targeted small bodies, landing on small bodies must be done autonomously using onboard sensors and algorithms. Machine vision technologies are being developed that can recognize landmarks for navigation, estimate spacecraft motion by tracking surface features, and identify safe landing sites [18]. These technologies are being combined with guidance navigation and control algorithms on existing real-time testbeds to provide an integrated capability for precise navigation, landing, and hazard avoidance.

Hazard Detection

A lander must be able to detect safe landing sites from hazardous ones. We use motion stereo vision to generate dense 3D topographic maps of a small body surface from monocular image streams. Image-based motion estimation is applied to determine the spacecraft motion between each frame, then each image is rectified to a fixed plane and stereo vision techniques are used to match image pixels to obtain pixel level depth estimates [19]. Terrain hazards are then extracted from the surface map, Figure 7.

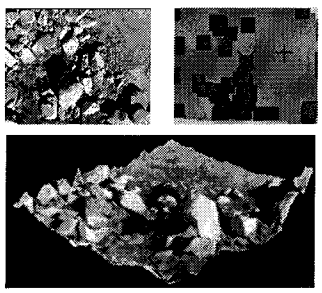

Figure 7: 3D surface reconstruction and hazard map.

Position Estimation

Our research combines two complementary methods for position estimation: feature tracking and landmark recognition. Feature tracking detects and tracks image features through a sequence of images enabling the six degree-of-freedom (DoF) relative motion of the spacecraft to be determined for each frame.

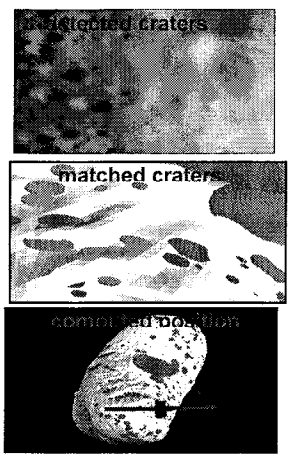

Figure 8: Position estimated from recognized landmarks.

Landmark recognition first detects landmarks during 3-D modeling of the body and stores them in a database along with their positions. To estimate position, landmarks are detected and matched against the database to obtain the absolute position of the spacecraft relative to the body [20]. By combining the continuous updates of relative position from feature tracking with the occasional updates of absolute position from landmark recognition, continuous estimates of spacecraft position in absolute body centered coordinates can be obtained, Figure 8.

Autonomous Landing

This research will develop a closed-loop image-based safe and precise landing capability by integrating position estimation, hazard detection, and 3D surface reconstruction algorithms with an aerial test bed composed of a commercial model-helicopter chassis, onboard processing and multiple navigation sensors.

We now turn to another key element of autonomous systems – the capability of a system to determine its overall health, diagnose its internal faults, and either provide self-generated remedies or otherwise signal potential fixes. This capability is commonly referred to as integrated vehicle health management. Such a capability greatly increases the probability of mission success, as well as reducing the need for human involvement and intervention. In the next section a strategy to achieve such a system capability is described, as well as a planned flight demonstration.

Integrated Vehicle Health Management

One of the current approaches to achieving this important capability is the Livingstone model-based health management system. Livingstone is an advanced inference engine that uses a high-level declarative model of a physical device to monitor the state of that device, detect off-nominal behavior, isolate failures to individual components, and reason about alternative recovery actions. The key benefit provided by Livingstone is the use of a first-principles model that describes the behavior of each component within the device and the interactions between the components [5,6,7]. By reasoning generatively about the behavior of the device using the model, Livingstone is able to detect failures whenever a discrepancy occurs between the observations and predictions. In addition, Livingstone is able to use the same model to generate the most likely hypothesis that is consistent with the observations and to select the optimal reconfiguration action for recovering from the failure. Thus, with the use of a model of the device, Livingstone is able to reason about novel combinations of failures and avoids the need to develop mission-specific code that must pre-enumerate all of the various failure combinations that might occur. Furthermore, the models used by Livingstone are easy to update and maintain and can often be reused across missions, thus further reducing the software development costs while increasing the functionality provided. A more detailed description of the Livingstone system is provided in the following sections.

Modeling Paradigm

As mentioned above, Livingstone uses a high-level, compositional model to identify the components within the device and the relationships between the components. This model is used for prediction, fault detection, isolation and reconfiguration. A Livingstone model is comprised of a set of components and connections between these components. Each component is modeled using a set of discrete valued variables. For example, a valve might be modeled using the variables *flow-in, flow-out, pressure-in* and *pressure-out* with values such as *zero, low, nominal, high.* For each component, a set of *modes* is defined identifying both the nominal and failure modes of the device. For each mode, a set of constraints is specified that restrict the values of the component variables whenever the component is in that mode. Thus, a valve might be modeled using modes such as *open, closed, stuck-open* and *stuck-closed* where the model of the valve in the *open* mode might be:

$$flow\text{-}in = flow\text{-}out$$
$$pressure\text{-}in = pressure\text{-}out$$

In addition, to the description of the behavior of the device for each mode, the model also includes transitions between modes with guard conditions describing when the transition occurs along with relative probabilities on the likelihood of the transitions. These probabilities are used to provide information about the relative likelihood of various failures. Figure 9 shows how a valve model might be represented as a finite-state automaton in which the labels on the links correspond to device commands.

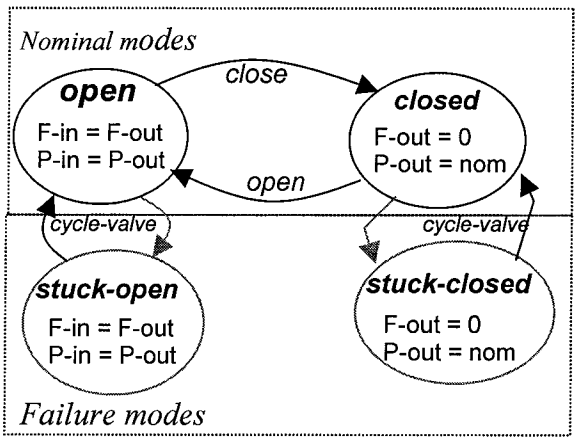

Figure 9. Valve model

One of the key benefits of this modeling paradigm is that the modeler is only responsible for describing the *local* behavior of each component and the relationships that exist between components. Livingstone then uses this specification to compose a larger, system model that can be used to reason about the *global* behavior of the entire system given the mode of each component. Furthermore, since the models are qualitative in nature it is often clear as to how to develop many of these models, even before the hardware design is complete.

Inference with Livingstone

Given a model of the form described in the preceding section, Livingstone performs two main tasks: 1) inferring the current state of the device given the limited available sensor information; and 2) identifying an optimal set of commands for system reconfiguration following a failure or external perturbation that transitions the system out of the desired state. At first glance, it might appear that the valve model described in the previous section is too simple to be of much use in performing these tasks. In practice, however, qualitative models of this nature have been found to be quite effective for detecting a wide range of likely failures. In fact, it is exactly these types of models that humans often use when reasoning about the current state of a device.

To estimate the current state of the system, Livingstone monitors the sequence of discrete commands that are issued to device or system. This allows the tracking of the expected state of the device and compares the predictions generated from its model against the observations received from the sensors. Once a discrepancy occurs, Livingstone performs a diagnosis by searching for the most likely set of component *mode assignments*[1] that are consistent with the observations. This is done using a search technique called *conflict-directed, best-first search,* developed within the model-based reasoning area of the artificial intelligence community. This search technique is able to efficiently search an exponentially large set of failure modes by focusing on the components whose state results in a conflict between the observations and the predictions. Within the Deep Space One experiment, for example, this search technique was able to identify the most likely component failure within a couple hundred milliseconds for most device failures.

Once the state of the system is identified, the same search techniques can then be used when reasoning about reconfiguration commands to identify the lowest cost set of commands that can be issued to transition the system into a state that satisfies the current operational goals provided by a higher level executive.

As a particular example of the application of the Livingstone architecture, a testbed experiment was planned that would allow flight testing of such an architecture on an experimental resuable launch vehicle. That application will be described in the following section.

The NITEX System

The NASA IVHM Technology Experiment for X-vehicles (NITEX) was selected to fly on the X-34 reusable launch vehicle being developed by Orbital Sciences Corporation. NITEX is being developed collaboratively between NASA Ames Research Center, NASA Glenn Research Center, and Kennedy Space Center. The X-34 vehicle is shown in Figure 10. The goal of the X-34 IVHM flight experiment is to advance the technology readiness levels of selected IVHM technologies within a flight environment and to begin the transition for these technologies from experiment status into accepted RLV baseline designs.

The experiment will monitor the X-34 vehicle throughout all mission phases using detailed diagnostic algorithms to detect degraded component performance as well as a system-level health monitoring system that integrates information from multiple components to perform real-time fault detection, isolation and recovery. In addition, the experiment will demonstrate the use of an advanced, user friendly ground station that combines information provided by the on-board IVHM software with information obtained while the vehicle was on the ground

[1] The *state* of the device is represented by identifying the current *mode* of each component in the model.

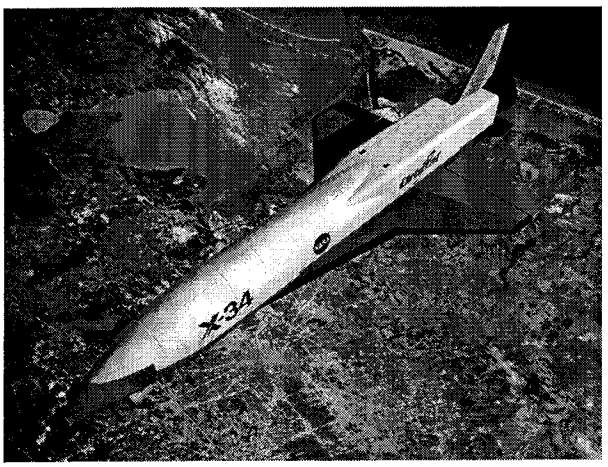

Figure 10. X-34 Flight Vehicle

to provide high-level status information on the health of the vehicle along with the ability to access more detailed information when required. The ground station will also provide justification for the inferences made by the IVHM system and alternative recovery recommendations following a failure while in flight. The focus of the experiment will be on the X-34's Main Propulsion Subsystem (MPS) including the Fastrac Engine and the Reaction Control System (RCS). The NITEX experiment architecture is shown in Figure 11.

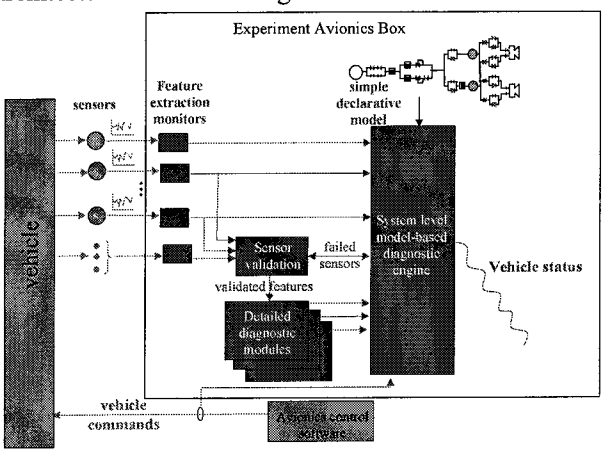

Figure 11. Overview of the NITEX Experimental Architecture

This experiment will monitor all the sensor data pertaining to this subsystem and track commands to determine the expected state of the device and ensure the vehicle is responding appropriately to the commands that have been issued. Sensor data from related subsystems (e.g., power) might also be monitored when appropriate to help isolate anomalies that are manifested within the MPS sensor suite. Selected components will be used to demonstrate the detailed diagnostic algorithms used to detect component degradation. The components used will be selected based upon information obtained from OSC and Marshall Space Flight Center (MSFC) regarding potential problems within the vehicle and the available sensor data. In addition, the experiment team will provide a ground-based monitoring station that monitors the vehicle during preflight and postflight processing, displays the results of the experiment while the vehicle is in flight, and provides advanced summarization and explanation capabilities to justify the inferences made by the experiment software.

A Multi-Agent Architecture

One reason to build a model-based discipline for the development of autonomous system software is to reduce the complexity and cost of achieving the same system functionality with respect to traditional software engineering methodologies. In the future, as our space exploration becomes more daring and more distant, the capabilities required from the explorer's controllers and their complexity is going to rise at an exponential pace. The limits of our current method of spacecraft operations have already been reached.

For example, the Mars Exploration Rovers (MER) mission will operate two rovers on the surface of Mars starting in 2004 with autonomy on board limited to collision avoidance during traversal and with ground operators intimately involved in all aspects of rover commanding and science observation planning. The traversal plan for MER calls for at most 1Km traversal and at most 12 targets for scientific observation per rover during the 90 days nominal mission duration. The next rover mission, currently slated for 2007, will require traversals in the order of 10Km and an increase in the number of scientific targets. Since the nominal mission duration is considered to be about 180 days, it is clear that the MER approach to mission operations will not suffice. Necessary autonomous capabilities include the ability of the rover to autonomously plan traversal beyond the horizon of its initial stationary panorama, the ability to reach a target location with accuracies of a few centimeters, and the ability to reliably position instruments carried by a deployment arm on exactly the same location on a target.

The development of an integrated and reliably software system for a highly autonomous mission requires overcoming several major challenges. The first challenge is achievement of a fully integrated approach to the development both of autonomy software and traditional control for mission software. At present, as demonstrated by the Remote Agent Experiment, autonomy software is still seen as a "add on" to traditional flight software. There are two problems with this approach. First, it makes the insertion of autonomy capabilities less attractive from

a cost perspective, since the implication is that one would still have to develop a traditional approach to spacecraft commanding and fault protection besides the autonomy software approach. Second, it prevents from spreading the benefits of the model-based approach to the development of traditional software. This is particularly important because the closer the software is to the hardware and the more difficult it is to obtain repeatable behaviors purely with extensive testing. Making the model of how the software behaves explicit allows the direct application of formal validation and verification techniques that significantly increase the reliability of mission-critical control loops.

One possible approach to overcoming these challenges, being investigated by the IDEA project, is to implement complex software control systems as a collection of interacting agents: i.e., software modules with an internal representation of the world with which they interact, their own internal functioning and the way they interact with the rest of the world. The model-based approach will be beneficial to model the communication protocols between agents and provide a way to validate a system in a compositional manner. The internal structure of each agent can then vary depending of what technology is used to implement it. Some agents will be implemented as traditional software and other using one of a variety of model-based software technologies. Rather than prescribing in detail how each individual module should operate internally, the compositional multi-agent approach will provide a standard that will facilitate the introduction of new autonomy technology in mission software, a situation that the current approach to flight software development makes extremely difficult.

Concluding Remarks

The coming decades will see many new opportunities to expand human presence in the solar system. As we penetrate deeper into space we must implement space exploration missions at lower cost, with greater safety, and achieve greater scientific return. Fortunately we are able to develop powerful new computer/information systems to meet these needs. To this end, NASA has embarked on the Computer, Information, and Communications Technology Program, which will research and develop new capabilities in the many areas of technology required for complex future missions, including automated reasoning, human-centered computing, intelligent use of data, and concepts for revolutionary computing, such as biomemetics and nanotechnology. Ultimately there will be a distinctly new balance of work between humans and intelligent machines, where tasks will reside with the entity having the best capability, irrespective of mechanism of origin. This will provide the total human-machine system with the capability to explore far beyond the limits of the present.

Acknowledgements

Portions of the work described in this paper were performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

References

Some of the following papers may be found on the World Wide Web at http://ic-www.arc.nasa.gov/ic/projects/mba/, and at http://www-aig.jpl.nasa.gov

1. R. Zubrin and R. Wagner. *The case for Mars: The plan to settle the Red Planet and why we must.* The Free Press, 1996.
2. S. J. Hoffman and D. I. Kaplan, editors. *Human Exploration of Mars: The Reference Mission of the NASA Mars Exploration Study Team.* NASA Special Publication 6107. July 1997.
3. S. Russel and P. Norvig *Artificial Intelligence: A Modern Approach*, MIT Press 1995.
4. D. E. Bernard et Al. Design of the Remote Agent Experiment for Spacecraft Autonomy. *Proceedings of IEEE Aero-98.*
5. J. de Kleer and B. C. Williams, Diagnosing Multiple Faults, *Artificial Intelligence*, Vol 32, Number 1, 1987.
6. J. de Kleer and B. C. Williams, Diagnosis With Behavioral Modes, *Proceedings of IJCAI-89*, 1989.
7. J. de Kleer and B. C. Williams, *Artificial Intelligence*, Volume 51, Elsevier, 1991.
8. B. C. Williams and P. P. Nayak. Immobile Robots: AI in the New Millennium. In *AI Magazine*, Fall 1996.
9. N. Muscettola. HSTS: Integrating planning and scheduling. In Mark Fox and Monte Zweben, editors, *Intelligent Scheduling*. Morgan Kaufmann, 1994.
10. A. Barrett, "Autonomy Architectures for a Constellation of Spacecraft," International Symposium on Artificial Intelligence Robotics and Automation in Space (ISAIRAS), Noordwijk, The Netherlands, June 1999.
11. P. Pirjanian, T. Huntsberger, A. Barrett, "Representation and Execution of Plan Sequences for Distributed Multi-Agent Systems," Proceedings of

the 2001 Intelligent Robots and Systems Conference, Maui, HI, November 2001.

12. A. Barrett, G. Rabideau, T. Estlin, S. Chien, "Coordinated Continual Planning Methods for Cooperating Rovers," Proceedings of the 2001 IEEE Aerospace Conference, Big Sky, MT, March 2001.

13. M.C. Burl, L. Asker, P. Smyth, U. Fayyad, P. Perona, J. Aubele, and L. Crumpler, "Learning to Recognize Volcanoes on Venus," *Machine Learning Journal,* April 1998.

14. M.C. Burl, W.J. Merline, E.B. Bierhaus, W. Colwell, C.R. Chapman, "Automated Detection of Craters and Other Geological Features *Intl Symp Artificial Intelligence Robotics & Automation in Space*, Montreal, Canada, June 2001

15. S. Chien, G. Rabideau, R. Knight, R. Sherwood, B. Engelhardt, D. Mutz, T. Estlin, B. Smith, F. Fisher, T. Barrett, G. Stebbins, D. Tran, "ASPEN - Automating Space Mission Operations using Automated Planning and Scheduling," *SpaceOps*, Toulouse, France, June 2000.

16. Interface & Control Systems, Spacecraft Command Language, http://www.sclrules.com.

17. T. Estlin, T. Mann, A. Gray, G. Rabideau, R. Castano, S. Chien and E. Mjolsness, "An Integrated System for Multi-Rover Scientific Exploration," Sixteenth National Conference of Artificial Intelligence (AAAI-99), Orlando, FL, July 1999.

18. A. Johnson, Y. Cheng, and L. Matthies, "Machine Vision for Autonomous Small Body Navigation," Proc. IEEE Aerospace Conf., March 2000.

19. Y. Cheng, A. Johnson, L. Matthies and A. Wolf "Passive Imaging Based Hazard Avoidance for Spacecraft Safe Landing," Proc. 6th ISAIRAS, June 2001.

20. A. Johnson, "Surface Landmark Selection and Matching in Natural Terrain," Proc. Computer Vision and Pattern Recognition, June 2000.

21. B. Williams and P. Nayak "A reactive planner for a model-based execution," *Proceedings of the 15th Joint International Conference on Artificial Intelligence (IJCAI '97), 1997.*